Running first task at HybriLIT cluster

sinfo[1] - getting information about batch system

```
[user@hydra ~]$ sinfo
PARTITION       AVAIL  TIMELIMIT       NODES      STATE NODELIST
interactive*    up     5:00:00             1      idle bladeo1
cpu             up     infinite            1      idle bladeo1
phi             up     infinite            2      idle blade[02-03]
gpu             up     infinite            2      idle blade[04-05]
```

- **interactive**: this queue should be used for testing. A task can take maximum 5 minutes. This is default queue (note * before name 'interactive');
- **cpu**: queue with standard cpu nodes;
- **phi**: queue with PHI processor nodes;
- **gpu**: queue with GPU nodes;

Quick look at module

Useful commands:

- **module avail** - list of available modules;
- **module add** <*module_name*> - add modules to the list of used modules;
- **module rm** <*module_name*> - remove a module from the list of used modules;
- **module switch** <*module_name_1*> <*module_name_2*> - replacement of the loaded module (the first) to the denoted module (the second);
- **module list** - list of loaded modules;
- **module show** <*module_name*> - description of the changes loadable modules;
- **module whatis** - list of modules and settings of the compiler, library, application, corresponding modules;

Running first job using batch mode

First we need to create file, use own editor, e.g. **script.sh** with the following content:

```
[user@hydra ~]$ cat script.sh
#!/bin/sh
hostname
```

Run batch mode:

```
[user@hydra ~]$ sbatch script.sh
Submitted batch job 601
[user@hydra ~]$ cat slurm-601.out
bladeo1.hydra.local
```

See output of **squeue** command:

```
JOBID   PARTITION    NAME        USER     ST      TIME        NODES  NODELIST(REASON)
  601       cpu      script.sh   hluser   R    1-01:52:41        1        bladeo1
```

[1] Note that queues like cpu, phi and gpu had infinite time for job execution, this setting can be changed in the future. Note that queue names can be also changed in the future. Please use sinfo to find out the latest setting.

Getting started

Copy work directory into home directory of user **hluser999** and change directory:

```
[user@hydra ~]$ cp -r /eos/hybrilit.jinr.ru/scratch/Tutorial_HSchool2014/ /eos/hybrilit.jinr.ru/user/h/hluser999/
[user@hydra ~]$ cd Tutorial_HSchool2014
[user@hydra ~]$ ls -l
drwx------ 14 hluser999 hybrilit 0 Sep  2 13:54 CUDA
drwx------  9  hluser999 hybrilit 0 Sep  2 13:54 MPI
drwx------  5  hluser999 hybrilit 4 Sep  2 13:54 OpenCL
drwx------ 13 hluser999 hybrilit 0 Sep  2 13:54 OpenMP
drwx------  6  hluser999 hybrilit 0 Sep  2 13:54 test
```

Compilation and run of MPI applications

Check and change directory:

```
[user@hydra ~]$ pwd
/eos/hybrilit.jinr.ru/user/h/hluser999/Tutorial_HSchool2014
[user@hydra ~]$ cd MPI
```

Add MPI modules:

```
[user@hydra ~]$ module add openmpi/1.8.1
```

- View example application (C language):

```
[user@hydra ~]$ cat hello.c
#include <stdio.h>
#include <mpi.h>
int main(int argc, char *argv[])
{
        int rank, size;
        MPI_Init (&argc, &argv);
        MPI_Comm_rank (MPI_COMM_WORLD, &rank);
        MPI_Comm_size (MPI_COMM_WORLD, &size);
        printf( "C: Hello world from process %d of %d\n", rank, size );
        MPI_Finalize();
        return 0;
}
```

Compiling of MPI application:

```
[user@hydra ~]$ mpicc hello.c
```

View **script_mpi**:

```
 [user@hydra ~]$ cat script_mpi
#!/bin/sh
#SBATCH -n 3
mpiexec ./a.out
```

where parameter **n** is the number of threads.

Run batch mode:

```
[user@hydra ~]$ sbatch script_mpi
Submitted batch job 9321
```

See output of **slurm-9321.out**:

```
[user@hydra ~]$ cat slurm-9321.out
C: Hello world from process 0 of 3
C: Hello world from process 1 of 3
C: Hello world from process 2 of 3
```

- Fortran language:

```
[user@hydra ~]$ cat hello.f
      include 'mpif.h'
      integer rank, size, ierr, tag, status(MPI_STATUS_SIZE)
      call MPI_INIT(ierror)
      call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierr)
      call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
      print*,'Fortran: Hello world',' from process ',rank,' of ',size
      call MPI_FINALIZE(ierr)
      end
```

Compiling of MPI application:

```
[user@hydra ~]$ mpif77 hello.f
```

Run batch mode:

```
[user@hydra ~]$ sbatch script_mpi
Submitted batch job 9322
```

See output of **slurm-9322.out**:

```
[user@hydra ~]$ cat slurm-9322.out
Fortran: Hello world from process  2  of  3
Fortran: Hello world from process  1  of  3
Fortran: Hello world from process  0  of  3
```

Compilation and run of OpenMP applications

Check and change directory:

```
[user@hydra ~]$ pwd
/eos/hybrilit.jinr.ru/user/h/hluser999/Tutorial_HSchool2014
[user@hydra ~]$ cd OpenMP
```

View example application:

```
[user@hydra ~]$ cat hellp.c
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
int main()
{
        int nthreads, tid;
        #pragma omp parallel private(nthreads, tid)
        {
                tid = omp_get_thread_num();
                printf("Hello World from thread = %d\n", tid);
                if (tid == 0)
                {
                        nthreads = omp_get_num_threads();
                        printf("Number of threads = %d\n", nthreads);
                }
        }
}
```

Compiling of OpenMP application:

```
[user@hydra ~]$ gcc hello.c -fopenmp
```

View **script_openmp**:

```
[user@hydra ~]$ cat script_openmp
#!/bin/sh
#SBATCH -n 4
./a.out
```

where parameter **n** is the number of threads.

Run batch mode:

```
[user@hydra ~]$ sbatch script_openmp
Submitted batch job 9323
```

See output of **slurm-9323.out**:

```
[user@hydra ~]$ cat slurm-9323.out
Hello World from thread = 0
Hello World from thread = 1
Hello World from thread = 2
Hello World from thread = 3
Number of threads = 4
```

Compilation and run of CUDA applications

Check and change directory:

```
[user@hydra ~]$ pwd
/eos/hybrilit.jinr.ru/user/h/hluser999/Tutorial_HSchool2014
[user@hydra ~]$ cd CUDA
```

Add CUDA modules:

```
[user@hydra ~]$ module add cuda-5.5-x86_64
```

- View first example application:

```
[user@hydra ~]$ cat test_CUDA_deviceInfo.cu
#include <stdio.h>
void CudaDevice(int deviceCount)
{
        size_t avail, total;
        cudaMemGetInfo(&avail, &total);
        for(int device = 0; device < deviceCount; device++)
        {
                cudaDeviceProp deviceProp;
                cudaGetDeviceProperties(&deviceProp, device);
                printf("CUDA Device number:    %d\n", device);
                printf("Name:              %s\n", deviceProp.name);
                printf("Total global memory:   %f Mb\n", deviceProp.totalGlobalMem/pow(2.0,20.0));
                printf("Total constant memory:     %d\n", deviceProp.totalConstMem);
                printf("Maximum threads per block:  %d\n", deviceProp.maxThreadsPerBlock);
                printf("=======================================================\n");
        }
}
int main()
{
        int deviceCount;
        int device;
        cudaGetDeviceCount(&deviceCount);
        printf("There are %d CUDA devices.\n", deviceCount);
        for(device = 0; device < deviceCount; device++);
        CudaDevice(device);
        return 0;
}
```

Compiling of CUDA application:

```
[user@hydra ~]$ nvcc -gencode=arch=compute_35,code=sm_35 test_CUDA_deviceInfo.cu -o test1
```

View **script_cuda**:

```
[user@hydra ~]$ cat script_cuda
#!/bin/sh
#SBATCH -p gpu
./test1
```

Run batch mode:

```
[user@hydra ~]$ sbatch script_cuda
Submitted batch job 9324
```

See output of **slurm-9324.out**:

```
[user@hydra ~]$ cat slurm-9324.out
There are 3 CUDA devices.
CUDA Device number:    0
Name:              Tesla K40s
Total global memory:   11519.562500 Mb
Total constant memory:     65536
Maximum threads per block:  1024
=======================================================
CUDA Device number:    1
Name:              Tesla K40s
Total global memory:   11519.562500 Mb
Total constant memory:     65536
Maximum threads per block:  1024
=======================================================
CUDA Device number:    2
Name:              Tesla K40s
Total global memory:   11519.562500 Mb
Total constant memory:     65536
Maximum threads per block:  1024
=======================================================
```

- Example of the second application with connection library:

Compiling of CUDA application:

```
[user@hydra ~]$ nvcc -gencode=arch=compute_35,code=sm_35 cuda_blas.cu –lcublas –o test1
```

Run batch mode:

```
[user@hydra ~]$ sbatch script_cuda
Submitted batch job 9325
```

See output of **slurm-9325.out**:

```
...After 10-20 seconds...

[user@hydra ~]$ cat slurm-9324.out
==== Element-wise vector multiplication by a scalar =====
Array size  NX = 1048576
GPU compute time_load data(msec): 5.27712
GPU compute time_cublas data(msec): 0.22877
CPU compute time_cpu (msec): 6.41411
============  Test result done on file Rezult_GPU.dat
============  Test result Vector for [0] and [1] elements
ix=0, iy =0, A_out = 0.0000000000000000e+00, A_input = 0.0000000000000000e+00
ix=1, iy =0, A_out = 9.4247779607693793e+00, A_input = 1.0000000000000000e+00
```

Compilation and run of OpenCL applications

Check and change directory:

```
[user@hydra ~]$ pwd
/eos/hybrilit.jinr.ru/user/h/hluser999/Tutorial_HSchool2014
[user@hydra ~]$ cd OpenCL
```

Add OpenCL modules:

```
[user@hydra ~]$ module add cuda-6.0-x86_64
```

Compiling of OpenCL application:

```
[user@hydra ~]$ nvcc helloWorld.c  –l OpenCL
```

View example application:

```
[user@hydra ~]$ cat helloWorld.c
```

View **script_opencl**:

```
[user@hydra ~]$ cat script_opencl
#!/bin/sh
#SBATCH -p gpu
./a.out
```

Run batch mode:

```
[user@hydra ~]$ sbatch script_opencl
Submitted batch job 9326
```

See output of **slurm-9326.out**:

```
[user@hydra ~]$ cat slurm-9326.out
Helo World! I am thread number 0.
Helo World! I am thread number 1.
…
Helo World! I am thread number 15.
```